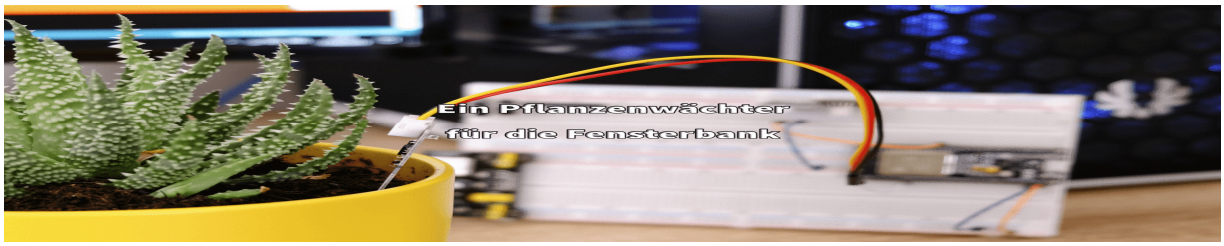


Pflanzenwächter für die Fensterbank Teil 2 – Eine eigene Handy App

Hallo und willkommen zum zweiten Teil unserer Pflanzenwächterreihe. Obwohl unser Pflanzenwächter schon ganz gut funktioniert, finde ich es doch als Maker etwas umständlich immer mal wieder nach der Led - „Ampel“ zu schauen, um zu sehen, ob meine Pflanze gegossen werden muss. Viel praktischer fände ich es, wenn ich jederzeit und überall einfach nur mal kurz auf das Handy schauen muss, um zu wissen, wie es der Pflanze zuhause geht. Daher werden wir uns in diesem Teil eine waschechte Handy APP für unseren Wächter bauen!

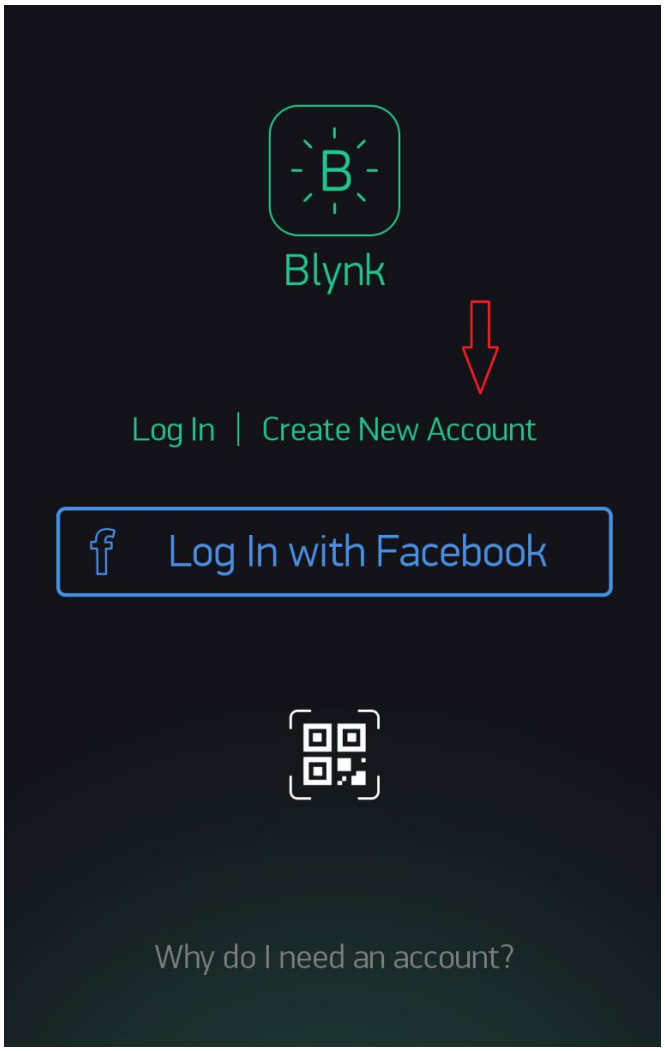
Doch bevor jetzt jemand jetzt denkt: Das geht doch nicht, das ist doch viel zu kompliziert, schauen wir uns doch mal zuvor genauer im App Store um. Es gibt nämlich eine bereits schon genau für unsere Zwecke konzipierte APP, die wir nur noch hier etwas anpassen müssen, da ein wenig Code zuzufügen müssen und dann das ganze zusammenfügen! Danach funktioniert unsere APP genauso wie wir es wollen.

Also los geht’s! Als ersten Schritt laden wir uns die App „Blynk“ aus dem App Store auf unser Handy. Die APP selbst ist kostenlos und kann für unseren, hier im zweiten Teil des Blogs, beschriebenen Anwendungsfall auch ohne weitere zusätzliche Kosten für die App betrieben werden.

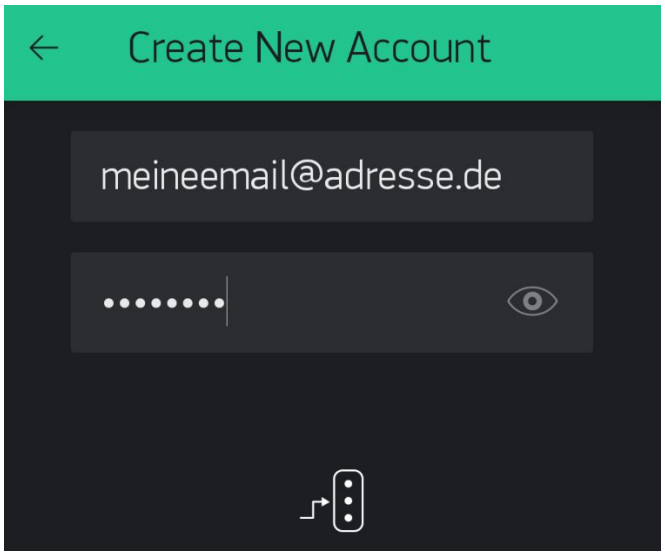
Für einen späteren weiteren Ausbau der App innerhalb der Blogreihe reichen die „2000 Energie“ Einheiten der kostenlosen Variante jedoch unter Umständen nicht mehr aus.

Es fallen aber unabhängig von der Ausbaustufe Kosten für die Datenübertragung vom Pflanzenwächter zum Handy an. Bitte berücksichtigt diese Kosten bei dem Nachbau des Projektes!

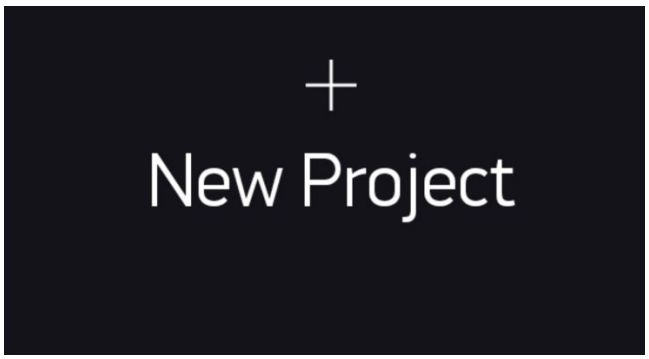
Nachdem wir also die App Blynk aus dem Store heruntergeladen haben und diese zum ersten Mal starten, müssen wir uns zuerst einmal einen Account anlegen. Dazu tippen wir auf „Create New Account“



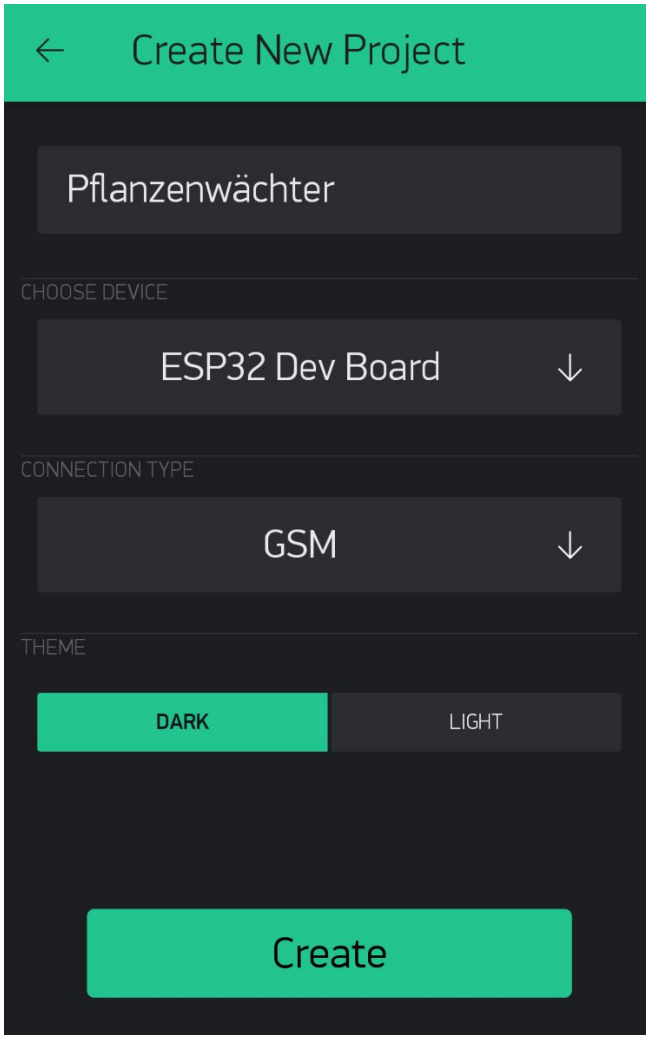
Wir registrieren uns mit der eigenen E-Mail-Adresse und vergeben ein Passwort:



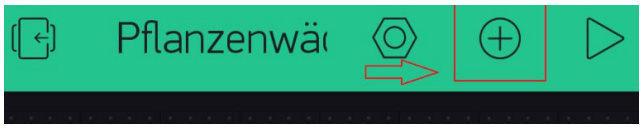
Danach klicken wir auf neues Projekt:



Wir landen in dem Dialog zur Anlage eines neuen Projektes. Hier müssen die Basisdaten des Projekts ein, wie z.B. den Projektnamen, unsere ESP32 Entwicklungsplattform und die gewünschte Verbindungsform. Es können verschiedene Verbindungsparameter mitgegeben werden, wie Z.B Bluetooth oder auch WLAN. Um jedoch auch Mobil unterwegs Daten empfangen zu können, muss der Verbindungstyp GSM gewählt werden. Nachfolgend sind die Einstellungen, die ich für mein Projekt gewählt habe, gezeigt:

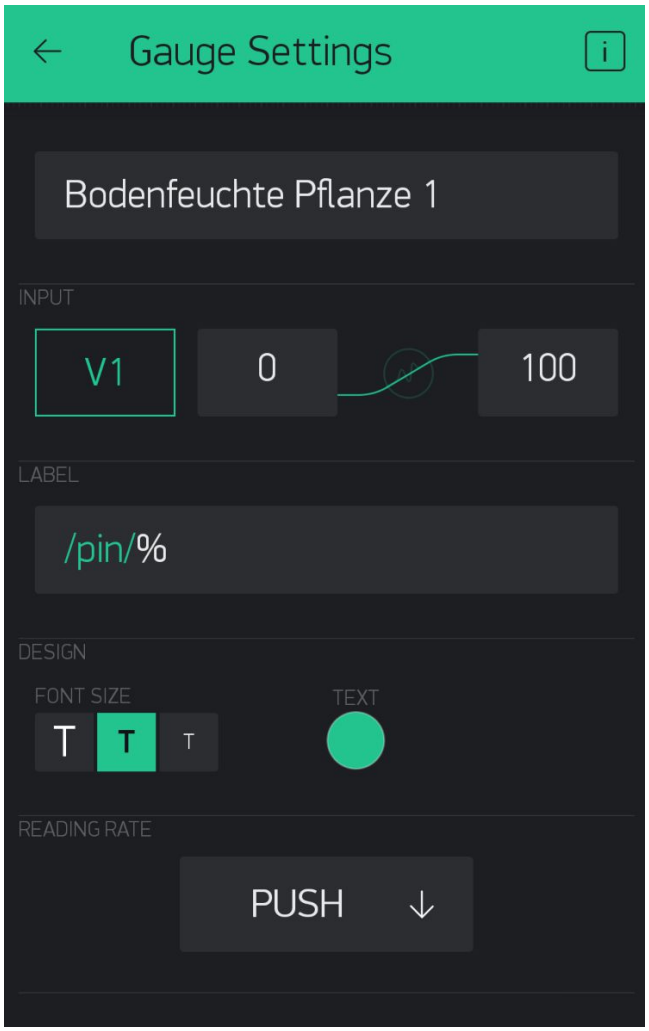


Wir bestätigen die Angaben mit „Create“ und landen in einem leeren Projekt. Wir fügen unser erstes Aktives Element in unsere App über das „Plus“ Symbol in der Titelleiste hinzu:

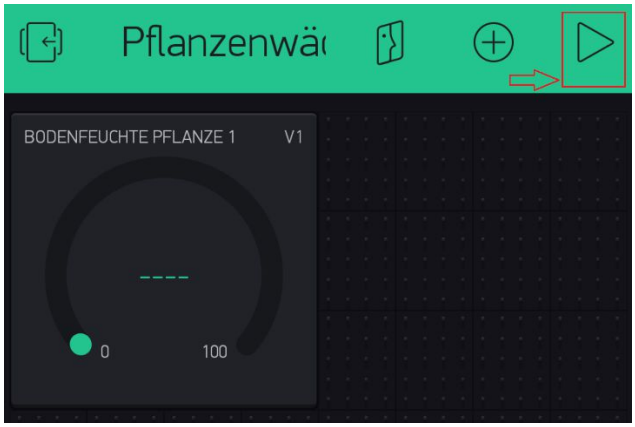


Wir wählen das Element „Gauge“ aus, und konfigurieren das Element. Zuerst vergeben wir einen NAMEN für das Element. Ich habe mich für den Namen „Bodenfeuchte Pflanze 1“ entschieden.

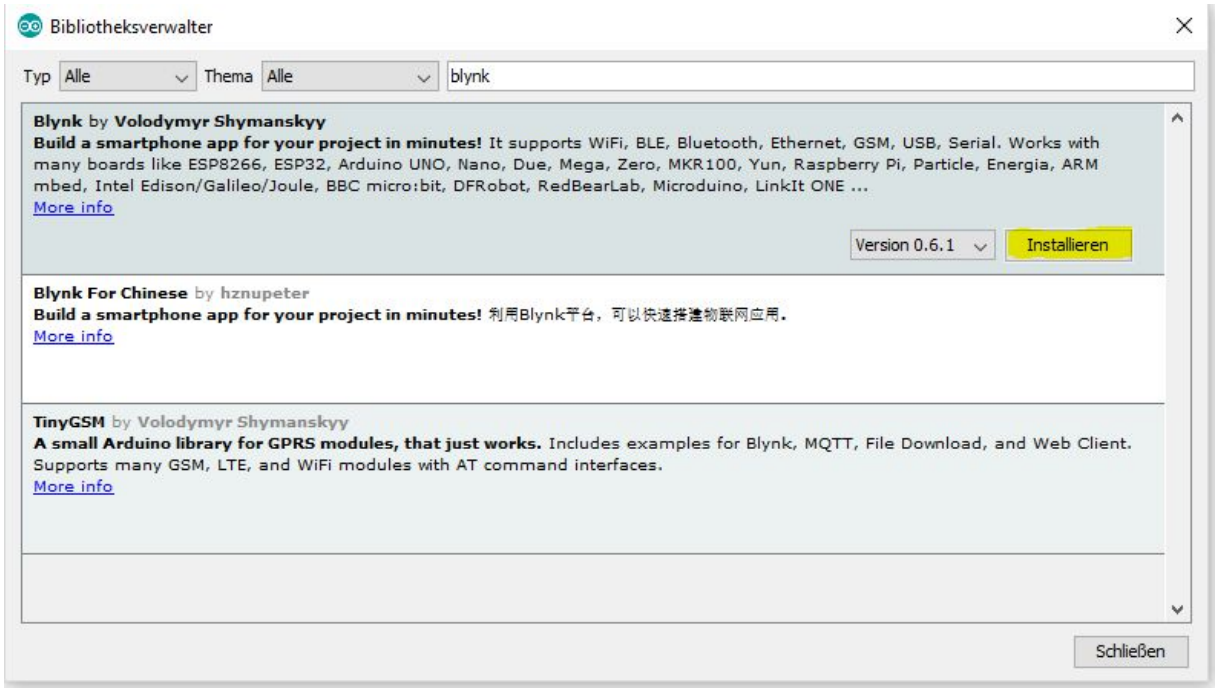
Danach wählen wir als Input Variable „V1“ aus, als Minimalwert 0 und als Maximalwert 100. Das Feld „LABEL“ ergänzen wir mit einem % und stellen als Reading Rate „Push“ ein. Bei dem Design können eine beliebige Farbe und eine beliebige Schriftgröße gewählt werden.



Wir bestätigen die Einstellungen mit dem Pfeil links, und landen dann wieder in der Hauptansicht der APP:



Die App können wir jetzt erst mal wieder schließen, denn wir müssen jetzt unsern ESP Code an unsere APP anpassen. Dazu installieren wir zunächst in der Arduino über den Bibliotheksverwalter die Blynk Bibliothek in der aktuellsten Version:



Danach fügen wir folgenden Code in die IDE ein:

```
#include <driver/adc.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>


// Portedefinierung Led's
#define LED_Rot    18   // Rote LED
#define LED_Gelb   14   // Gelbe LED
#define LED_Gruen  15   // Gruene LED

// LED PWM Einstellungen
#define PWMfreq 5000  // 5 Khz basisfrequenz
#define PWMledChannelA  0
#define PWMledChannelB  1
#define PWMledChannelC  2
#define PWMresolution  8 // 8 Bit Resolution

#define ADCAttenuation ADC_ATTEN_DB_11  //ADC_ATTEN_DB_11 = 0-3,6V  Dämpfung ADC
#define MoisureSens_Poll_Interval 60000  // Intervall zwischen zwei Bodenfeuchtemessungen in Millisekunden
#define MaxSensors 1

#define BLYNK_PRINT Serial
#define BLYNK_NO_BUILTIN
#define BLYNK_NO_FLOAT
//#define BLYNK_DEBUG

struct MoistureSensorCalibrationData
 {
   int Data[MaxSensors*2] = {1650,2840};  // Calibration Data für Feuchtigkeitssensor. Bitte Projekt Text beachten und Werte
ensprechend anpassen
 };

struct MoistureSensorData
 {
   byte Percent[MaxSensors] = {0};      // Feuchtigkeitssensordaten in Prozent
   byte Old_Percent[MaxSensors] = {0};   // Vorherige _ Feuchtigkeitssensordaten in Prozent (Zweck: DatenMenge einsparen.)
 };

//Global Variables
char auth[] = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"; // Hier lt. Anleitung Auth Token dere Blynk App eintragen (E-Mail).

// Deine WiFi Zugangsdaten.
char ssid[] = "WLANSSID";
char pass[] = "XXXXXXXX";  // Set Password to "" for open networks.

MoistureSensorCalibrationData MCalib;
MoistureSensorData MMeasure;
byte AttachedMoistureSensors; // Detected Active Moisture Sensors (Count)
unsigned long Moisure_ServiceCall_Handler = 0;  // Delay Variable for Delay between Moisure Readings

void setup() {
 pinMode(LED_Rot,OUTPUT);
 pinMode(LED_Gelb,OUTPUT);
 pinMode(LED_Gruen,OUTPUT);
 Serial.begin(115200);   // initialize serial communication at 115200 bits per second:
 ledcSetup(PWMledChannelA, PWMfreq, PWMresolution);
 ledcSetup(PWMledChannelB, PWMfreq, PWMresolution);
 ledcSetup(PWMledChannelC, PWMfreq, PWMresolution);
 ledcAttachPin(LED_Rot, PWMledChannelA);   // attach the channel to the GPIO to be controlled
 ledcAttachPin(LED_Gelb, PWMledChannelB);
 ledcAttachPin(LED_Gruen, PWMledChannelC);
 SetLedConfig(255,255,255);
 Serial.println(F("Systemkonfiguration:"));
 AttachedMoistureSensors = DetectMoistureSensors();
 Serial.print(AttachedMoistureSensors);
 Serial.println(F(" Bodenfeuchtigkeitsensor(en)"));
 Serial.print(F("Verbindung zu WLAN"));
```

```
 delay(500);
 Blynk.begin(auth, ssid, pass);  // Initalize WiFi-Connection over Blync Library
 Serial.println(F(" erfolgreich."));
 SetLedConfig(0,0,0);
 Run_MoistureSensors(true);
}

byte DetectMoistureSensors ()
 {
 #define MinSensorValue 100
 byte Detected = 0;
 for (int i = 0;i < MaxSensors;i++)
  {
  int MSensorRawValue = ReadMoistureSensor_Raw_Val(i);
  if ( MSensorRawValue > MinSensorValue) { Detected++; } else {break;}
  }
 if (Detected < 1)
  {
    Serial.println(F("Keine Bodenfeuchtigkeitssesoren erkannt. System angehalten."));
    esp_deep_sleep_start();
    while(1) {}
  }
 return Detected;
 }

bool SetLedConfig(byte Red,byte yellow,byte green)
{
ledcWrite(PWMledChannelA, Red); // Rote LED
ledcWrite(PWMledChannelB, yellow); // Gelbe LED
ledcWrite(PWMledChannelC, green); // Gruene LED
return true;
}

int ReadMoistureSensor_Raw_Val(byte Sensor)
 {
  int ReturnValue,i;
  long sum = 0;
  #define NUM_READS 6
  adc1_config_width(ADC_WIDTH_BIT_12);   //Range 0-4095
  switch (Sensor)
   {
   case 0:
    {
    adc1_config_channel_atten(ADC1_CHANNEL_0,ADCAttenuation);
    for (i = 0; i < NUM_READS; i++){  // Averaging algorithm
    sum += adc1_get_raw( ADC1_CHANNEL_0 ); //Read analog
    }
    ReturnValue = sum / NUM_READS;
    break;
    }
   }

 return ReturnValue;
 }

bool Get_Moisture_DatainPercent()
{
 bool ReadisValid = true;
 for (int i = 0;i < AttachedMoistureSensors;i++)
 {
 if ((MCalib.Data[i] == 0) || (MCalib.Data[i+1] == 0)) // MinADC Value maxADC ADC Value
  {
  ReadisValid = false;
  return ReadisValid;
  }
 int RawMoistureValue= ReadMoistureSensor_Raw_Val(i);
 RawMoistureValue= MCalib.Data[i+1] - RawMoistureValue;
 RawMoistureValue=MCalib.Data[i] + RawMoistureValue;
 //Serial.println(MCalib.Data[i]);
 //Serial.println(MCalib.Data[i+1]);
 //Serial.println(RawMoistureValue);
 MMeasure.Percent[i] = map(RawMoistureValue,MCalib.Data[i],MCalib.Data[i+1], 0, 100);
 if (MMeasure.Percent[i] > 100 ) { ReadisValid = false; }
```

```
  }
 return ReadisValid;
 }

void Run_MoistureSensors (bool Init)   // HauptFunktion zum Betrieb der Bodenfeuchtesensoren
{
byte MinSensValue = 100;
if ((millis() - Moisure_ServiceCall_Handler >= MoisureSens_Poll_Interval) | (Init))
  {
   Moisure_ServiceCall_Handler = millis();
   if (Get_Moisture_DatainPercent())  // Gültige aktuelle Daten Empfangen
   {
     for (int i = 0;i < AttachedMoistureSensors;i++)
      {
       if (MMeasure.Percent[i] != MMeasure.Old_Percent[i])
        {
          MMeasure.Old_Percent[i] = MMeasure.Percent[i];
          if (MMeasure.Percent[i] < MinSensValue ) { MinSensValue = MMeasure.Percent[i]; };
          Serial.print(F("Feuchtigkeitswert Sensor 1 in Prozent :"));
          Serial.print(MMeasure.Percent[0]);
          Serial.println(F(" %"));
          if (i == 0)  {Blynk.virtualWrite(V1,MMeasure.Percent[i]);  } // Aktualisiere Handywerte
        }
      }
     if (MMeasure.Percent[0] > 50)
      { SetLedConfig(0,0,20); }
     else if (MMeasure.Percent[0] > 10)
      {
       SetLedConfig(0,255,0);
       }
     else
      { SetLedConfig(255,0,0); }
   }
   else  // Keine gültigen Daten empfangen
   {
    Serial.print(F("Bodenfeuchtigkeitssensor nicht kalibiert.Bitte kalibieren. RohDaten des Sensors 1:"));
    Serial.println(ReadMoistureSensor_Raw_Val(0));
    SetLedConfig(255,255,255);
   }
  }
}

// Main Loop
void loop()
{
 Run_MoistureSensors(false);
 Blynk.run();   // Execute Blync Basic- Functions
}
```

Jetzt müssen wir noch unseren Code ein wenig anpassen. Zum einem muss im Code die WLAN Credentials (SSID und Passwort) an das heimische WLAN angepasst werden:

```
// Deine WiFi Zugangsdaten.
char ssid[] = "WLANSSID";
char pass[] = "XXXXXXXX";  // Set Password to "" for open networks.
```

zum anderen haben wir bei der Anlage unseres Projektes in der AP ein Authenticator Token zugesendet bekommen:

Auth Token for Pflanzenwächter project and device Pflanzenwächter

Blynk <dispatcher@blynk.io>
an mich

Auth Token : [redacted]

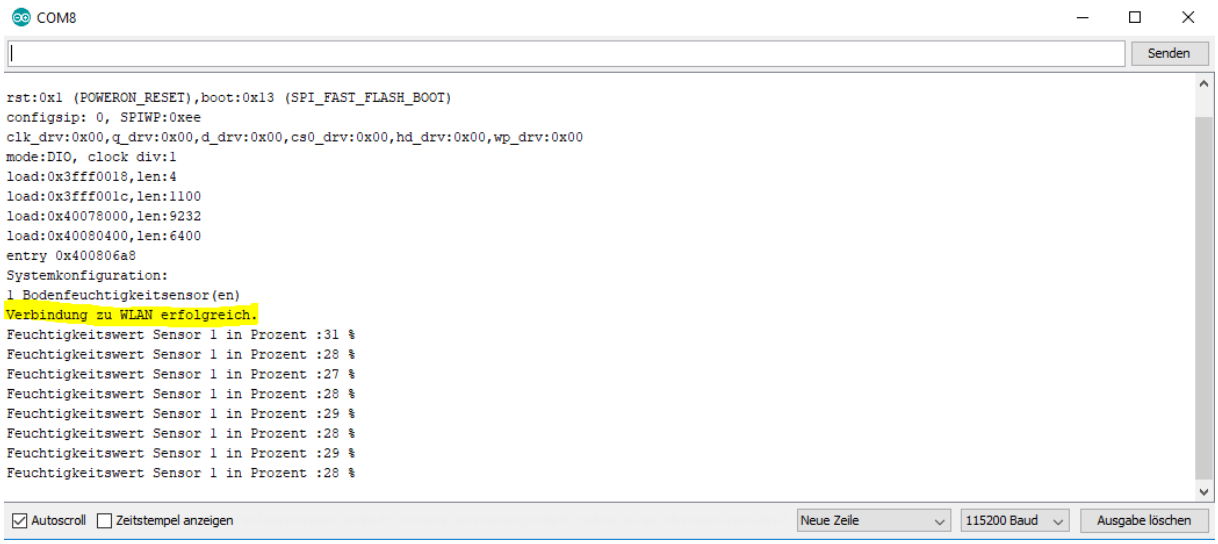
Happy Blynking!
-
Getting Started Guide -> https://www.blynk.cc/getting-started
Documentation -> http://docs.blynk.cc/
Sketch generator -> https://examples.blynk.cc/

Latest Blynk library -> https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip
Latest Blynk server -> https://github.com/blynkkk/blynk-server/releases/download/v0.41.5/server-0.41.5.jar
-
https://www.blynk.cc
twitter.com/blynk_app
www.facebook.com/blynkapp

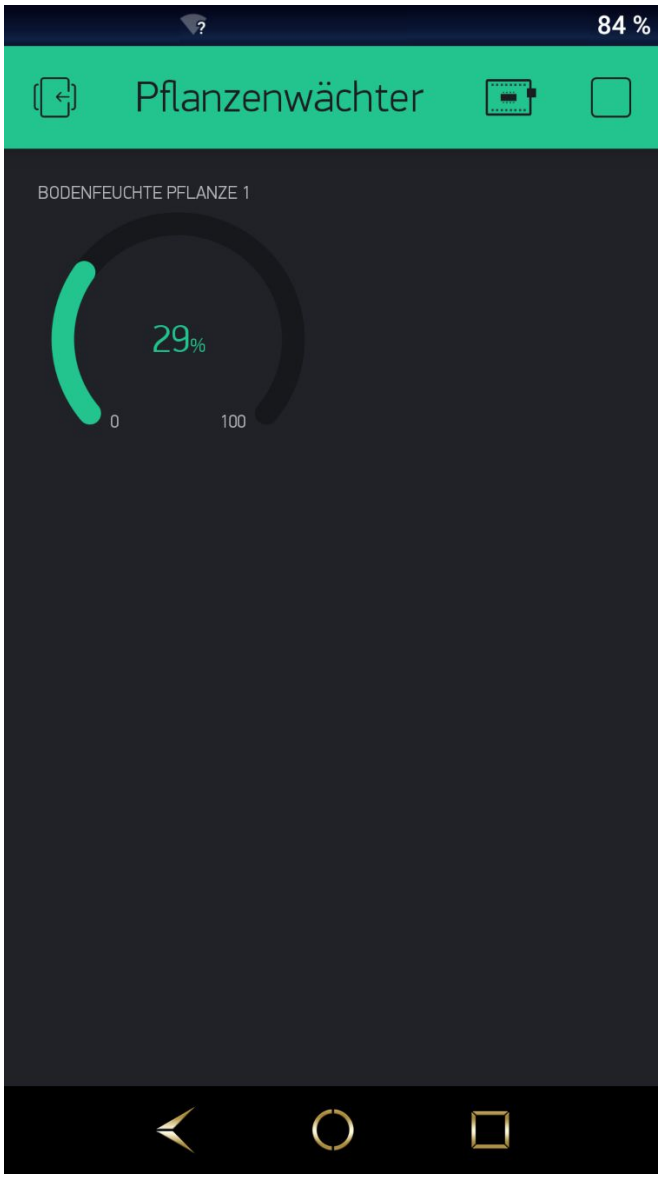
Dieses Token tragen wir im Code in der Zeile:

```
//Global Variables
char auth[] = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"; // Hier lt. Anleitung Auth Token dere Blynk App eintragen (E-Mail).
```

ein und kompilieren den Code. Anschließend laden wir Ihn aus unseren ESP hoch. Wir sollten nun auf der seriellen Schnittstelle eine ähnliche Ausgabe wie diese bekommen:



Nun kann die Blynk App auf dem Handy gestartet werden. Nach der Anmeldung und starten der neu erstellten Anwendung zeigt uns die App den aktuellen Feuchtigkeitswert des Bodenfeuchtesensors in Prozent an


84 %
Pflanzenwächter
BODENFEUCHTE PFLANZE 1
29%
0
100

Weitere Informationen über die Blynk APP und deren Verwendung in Controllern findest du unter

- Blynk Einführung -> https://www.blynk.cc/getting-started
- Dokumentation -> http://docs.blynk.cc/
- Sketch Generator -> https://examples.blynk.cc/
- Aktuellste Blynk Bibliothek -> https://github.com/blynkkk/blynk-library/releases/download/v0.6.1/Blynk_Release_v0.6.1.zip
- Aktuellster Blynk server -> https://github.com/blynkkk/blynk-server/releases/download/v0.41.5/server-0.41.5.jar
- Blynk Startseite -> https://www.blynk.cc

Ich wünsche viel Spaß beim Nachbauen, und bis zum nächsten Mal.